(12) UK Patent Application (19) GB (11) 2 377 138 (13) A

(43) Date of A Publication 31.12.2002

(21) Application No 0115835.1

(22) Date of Filing 28.06.2001

(51) INT CL[7]
G06F 13/40 , H04L 12/42

(52) UK CL (Edition T)
H4P PPBC

(56) Documents Cited
GB 2188216 A
JP 100228445 A
US 4884192 A
EP 1069509 A2
US 4982400 A
US 4641276 A

(58) Field of Search
UK CL (Edition T) G4A AFGDC, H4P PPBC
INT CL[7] G06F 13/37 13/40, H04L 12/42 12/427 12/43 12/433
Other: Online: EPODOC, JAPIO, WPI

(71) Applicant(s)
Telefonaktiebolaget L M Ericsson
(Incorporated in Sweden)
SE-126 25, Stockholm, Sweden

(72) Inventor(s)
Rowan Nigel Naylor

(74) Agent and/or Address for Service
Haseltine Lake & Co
Imperial House, 15-19 Kingsway,
LONDON, WC2B 6UD, United Kingdom

(54) Abstract Title
Ring Bus Structure For System On Chip Integrated Circuits

(57) A bus structure for an integrated circuit designed to accommodate the types of operation used in telecommunications systems and to reduce the required bus frequency for a given system whilst also reducing power and processing power. The bus comprises a plurality of interface modules which are connected in series to form the bus structure, where each module is operable to transfer data to at least one adjacent module in the series, and including an interface unit for connection with a host peripheral, and a storage element for storing data.

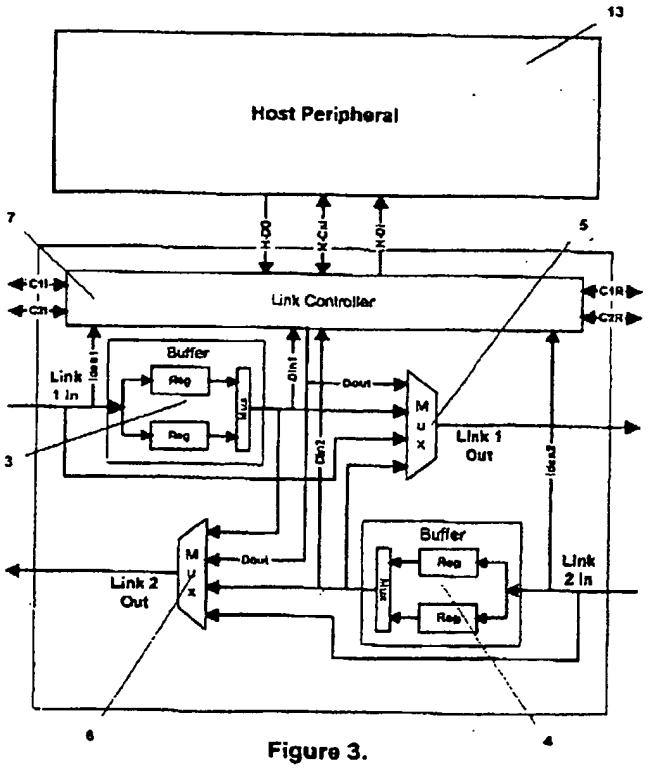


Figure 3.

GB 2 377 138 A

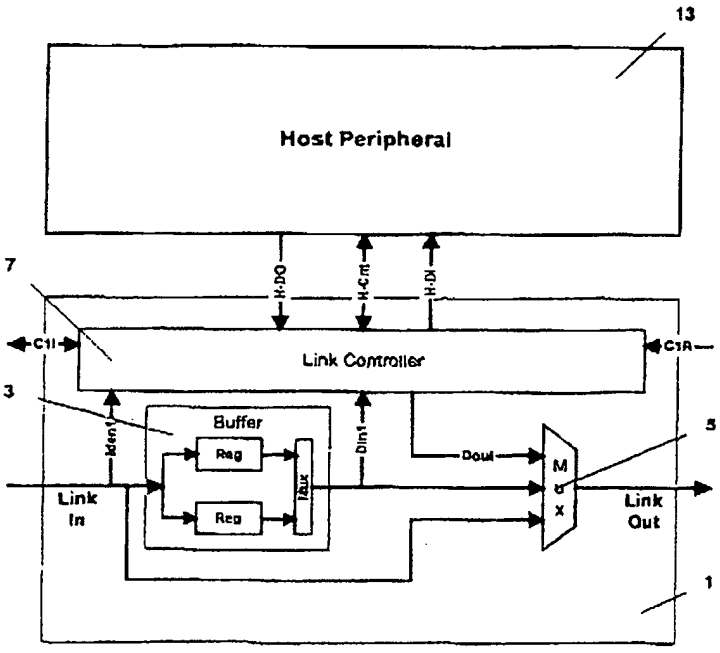


Figure 1.

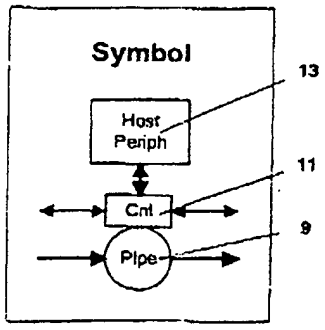


Figure 2.

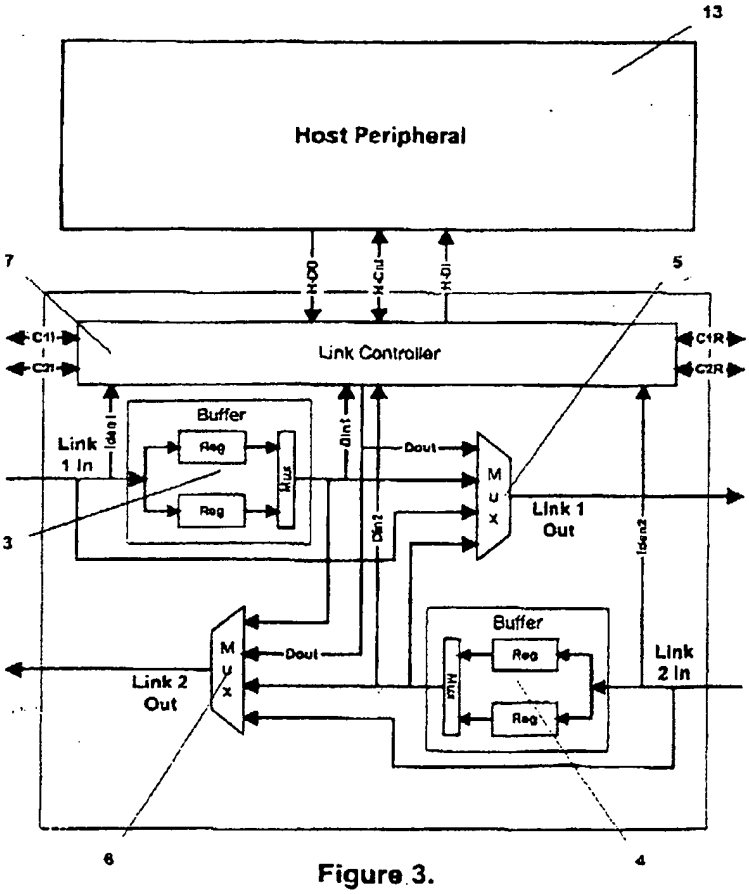


Figure 3.

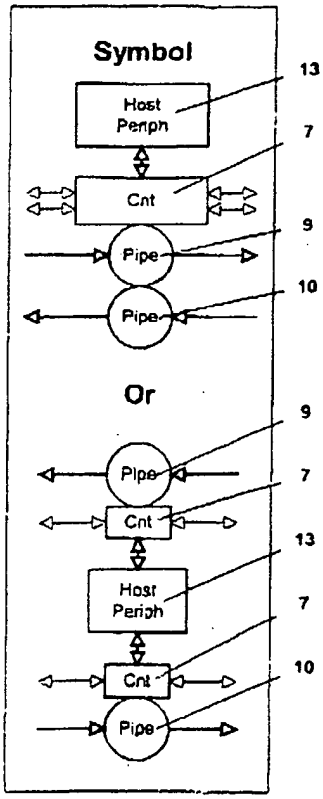
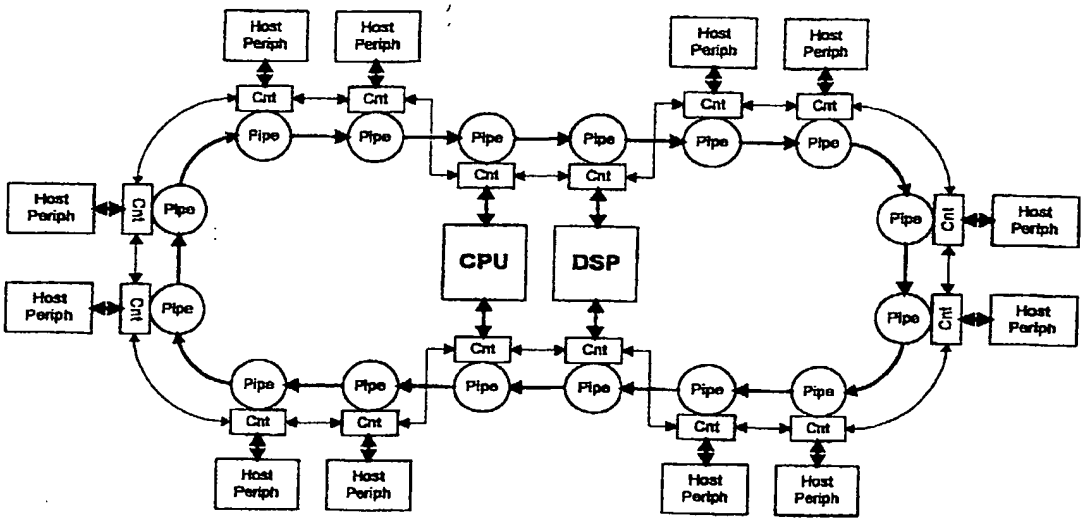
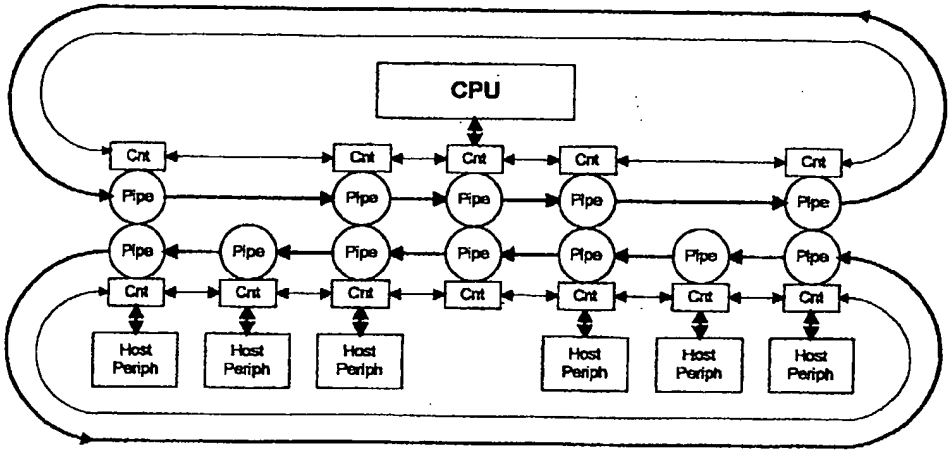


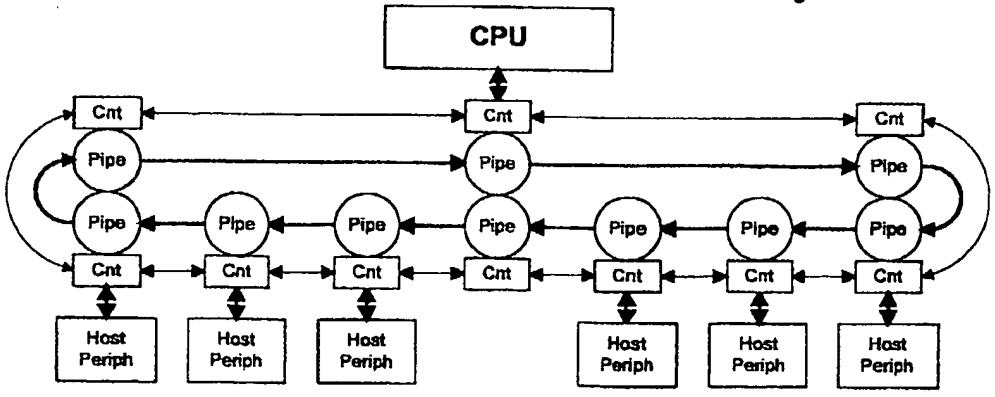
Figure 4.



Dual Processor Single Pipe Ring Example Figure 5



Dual Port - Dual Ring Bus Architecture Figure 6



Chain Ring Bus Architecture Figure 7

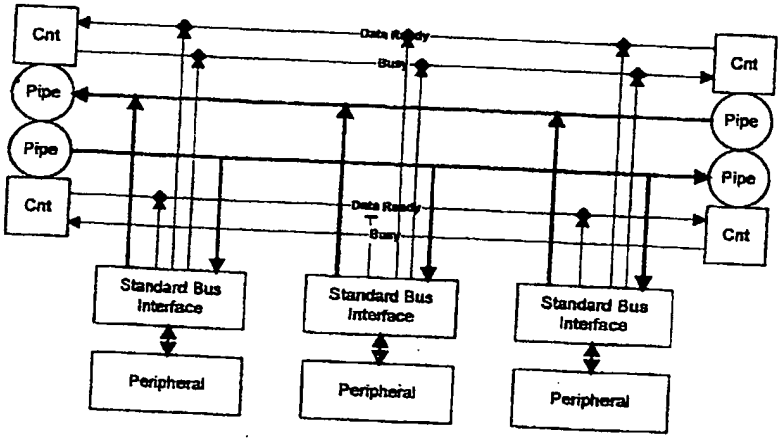

Cnt
Pipe
Pipe
Cnt
Data Ready
Busy
Cnt
Pipe
Pipe
Cnt
Data Ready
Busy
Standard Bus Interface
Standard Bus Interface
Standard Bus Interface
Peripheral
Peripheral
Peripheral

Figure 8

2377138

BUS ARCHITECTURES

The present invention relates to bus architectures for use in integrated circuits.

Background of the invention

The majority of modern 'System on Chip' (SoC) designs use known bus architectures and methodologies to implement processor driven systems. These are typically based on a master slave approach with a master unit (eg. a processor) controlling all operations on the bus and accessing slave units using memory mapped addressing. An address space is viewed as a sequential set of memory locations in which each slave unit (eg. peripheral) is assigned a collection of addresses.

The master unit controls all transfers between slave units by reading from one slave unit into a buffer and then writing to another slave unit from that buffer. This requires a two stage transfer process.

This process is simplified in some known systems by using a 'Direct Memory Access' (DMA) unit that executes this operation as back to back memory accesses. This is more efficient than using the processor itself as it can be hardwired and does not require software. However, the bus must be a multi-master design to allow control of the bus to be passed between processor and DMA unit.

In either case the maximum transfer rate of data is half the maximum rate sustainable by the bus, because there are two transfer actions required per data transfer.

In the telecommunications industry the design of a system architecture typically uses sequential sets of processes to implement a function, such as a modem. Data is passed along a series of processes in sequence,

each process modifying the data and passing it on to the next process in the series.

This can be implemented without a bus structure as such by hardwiring one hardware process block to the next. However, increasingly some of these processes are being implemented in software for processing by a DSP or CPU. In addition, in order to increase the flexibility and testability of the system it is necessary for the DSP/CPU to have access to the input and output of all process blocks in the sequence.

In the traditional bus architecture this requires each block to be memory mapped to the DSP/CPU bus, the DSP/CPU is then responsible for the passing of data between the blocks. This can be supported by a DMA unit, that is programmed by the DSP/CPU, as described above.

As the required performance of the system increases, the effect on the system bus is magnified, and so the required bandwidth reaches the limits of the technology.

For example, consider a modem implementation consisting of 5 sequential process blocks, if each block has the same data input rate and data output rate of X bytes/second then the minimum total data rate required is 10 X bytes/second.

As the data throughput of the modem increases the required bus performance has to increase by a factor of 10 which can be difficult to achieve.

In addition, the movement of data by the processor consumes substantial processing power and electrical power, both of which are at a premium ideally and need to be minimised, particularly in mobile and handheld systems.

Summary of the present invention

The bus architecture of the present invention is designed to accommodate the types of operation used in telecommunication systems and to reduce the required bus frequency for a given system while also reducing power and processing power.

It is emphasised that the term "comprises" or "comprising" is used in this specification to specify the presence of stated features, integers, steps or components, but does not preclude the addition of one or more further features, integers, steps or components, or groups thereof.

Brief description of the drawings

Figures 1 and 2 illustrate a single port bus module embodying one aspect of the present invention;

Figures 3 and 4 illustrate a dual port bus module embodying another aspect of the present invention.

Figures 5, 6 and 7 illustrate respective example bus architectures embodying the modules of Figure 1 to 4; and

Figure 8 illustrates a module of Figures 1 to 4 interfacing with a known bus interface.

Detailed description of the preferred embodiments

A single port module 1 embodying one aspect of the present invention is shown in schematic form in Figure 1 and comprises a buffer memory element 3 and a multiplexer 5, both of which are controlled by a link controller 7. The module is illustrated connected with a host peripheral 2. The link controller 7 communicates with the host peripheral 2. Figure 2 shows a symbol representation of the module of Figure 1, the buffer 3 and multiplexer 5 are represented by a pipe 9 and the link controller by unit 11. As in

Figure 1, the module 1 of Figure 1, the module 1 of Figure 2 is shown connected with a host peripheral unit 2. Data is received by the module 1 in packetised form. A data consists of one or more words, each word containing one or more data bits. Incoming data packets are presented to the link controller 7, to the buffer 3 and to the multiplexer 5. The link controller 7 examines the incoming data packets to determine what action is needed. Each data packet contains a header portion identifying the packet, and a data payload which contains the data to be transferred.

The data packet header is examined by the link controller in order to determine the action required.

Incoming data packets are always loaded into the buffer 3 when the buffer 3 is free. The multiplexer 5 is connected to receive the output from the buffer 3, data from host peripheral 2 (via the link controller 7) and the data input to the module 1. The data output from the buffer 3, can also be routed via the link controller 7 to the host peripheral 13.

The link controller 7 determines if a data packet is intended for the host 2 and routes the data packet to it if that is the case. The link controller 7 also controls the multiplexer 5 and depending on the identity of the data packet will output the data packet from the module. Modules are connected together to form a bus structure, as will be described in more detail below. If the host 2 needs to output data packets to the bus then the link controller 7 causes the multiplexer 5 to output the host data packet to the next module in the bus. The data packet can be a priority transfer so that the link controller 7 routes the data input directly to the data output of the module.

The single port module shown in Figures 1 and 2 is a uni-directional device. A bi-directional dual port

module is illustrated in Figures 3 and 4, and contains two single port modules as described above. The directions of data in the bus for each port are generally opposite to one another. In addition a link is provided from the output of the buffer 3 in one module to the multiplexer in the other module. This allows a data packet from one path of the bus to the other path.

The link controller 7 interfaces to adjacent modules in the bus structure. The link controller 7 controls handshaking signals that transfers data packets from one module to the next. In the dual ring configuration using the dual port module the two rings preferably operate independently of one another. The transfer between modules can be either asynchronous or synchronous as required by the system design; the basic operation is the same.

Data packets are transferred along the bus from module to module.

The bus architecture can take the form of a ring when the last link in a chain of links is attached to the first in the chain; this allows data to be transferred between all links in the chain.

In the dual port link architecture a ring can be formed in a chain by linking the two ports of both the first and last link in the chain or a dual ring and 'mobius' ring by joining the last and first links in the chain.

Figures 5, 6 and 7 illustrate example bus architectures using the modules of Figures 1 to 4.

At any given time, the maximum data rate on the bus will be equal to the speed of the slowest link multiplied by the number of buffers in the ring. For example if there are 10 links in the ring, each with one word of buffering, and a transfer rate of 1M

packets per second, then the maximum bus data rate is 10M packets/second.

Each module presents the same single load to the previous module in the series forming the bus structure. Therefore as modules are added to the bus the overall loading on most of the existing modules is unaffected. Also, the speed of the bus is determined by the aggregate loading of the modules, which can be faster than in known bus architectures because of the lower loading on a given module.

The transfer of data packets between any two modules depends on the number of other modules between them. This is a latency that needs to be taken into account when setting up the bus structure. Given that this bus architecture is designed to service object/message based communication the latency can be considered to affect only the first word of a transfer; all subsequent words arrive at the destination at full speed since they would have the same latency. As a result the effective data rate is the total time, (latency time plus data transfer time) divided by the number of transfers. The larger the data packets transferred the lower the aggregate effect of the latency.

As described above, transfer of data packets between modules is controlled by the Link Controller 7 (LC) of each module. The link controller 7 is responsible for controlling the transfer of data packets into the buffer 3 and the subsequent use of the output multiplexer. The LC 7 also controls the transfer of data packets to and from the host peripheral using the link interface.

The transfer control is also basically the same for both asynchronous and synchronous operation, the only difference being that in the synchronous case all actions take place with respect to a clock edge, and in

the asynchronous case they take place as quickly as possible.

A two-signal handshake is used for the transfer and moves one data packet from one module to the next module.

An overall data transfer may contain multiple data packets that are sent sequentially. In such a case the same transfer mechanism operates as for single data packet transfers except the header structure is modified. The first packet of the transfer contains a full header and all subsequent data packets contain only the link identifier with the last word containing an indicator showing end of package.

The link controller at a destination for the payload ensures that the package is reconstructed. Also the sequence of transfers are treated as continuous in that no host peripheral between source and destination is allowed to insert a package midway through the sequence.

If the receiving link is busy, either the next link ahead to it is busy or the host is inserting a payload then the transmitting link is held off for as long as required.

The buffer 3 of the link interface may be capable of storing more than data packet, using a FIFO structure. This can be used to reduce the delays incurred in inserting data packets.

In a ring structure there exists the possibility of a lock-up in which a delay forces the busy signal to be active in all modules, that is the delay in a module accepting data ripples back around the ring till it is itself halted by a busy signal ahead.

This can occur if all modules already contain payloads filling their buffers 3 but not directed at them and then another payload is inserted by a host.

This is prevented by giving the contents of the buffer 3 priority over data to be inserted by the host. The host can only insert data if its buffer 3 is not full.

This is tempered with the ability to 'Force' data onto the bus by temporarily removing the contents of the full buffer 3 associated with the link doing the forcing.

This mechanism allows a host with priority payload to accept the payload aimed at another link and substitute its own. The intercepted payload can then be either discarded or sent when capacity is available.

The host peripheral attached to the intercepting interface makes the choice based on how much local memory is available. In the case of limited local memory then the data would be discarded. The receiving peripheral using the header information would detect the missing data and request retransmission of the lost data.

However, the method uses the buffer in the interface and then signals controlling the transfer are caused to ripple back to the transmitting peripheral to halt its data transfer.

The mechanism works by double buffering incoming data on a halt. The interface that is inserting data does not pass on the present data stored in the buffer but instead accepts the next data packet into a second buffer register while indicating to the previous interface that it is busy.

The previous interface stops transfer of the new data in its buffer but accepts data from its previous interface into its second register while indicating it is busy. This is repeated back along the chain to the source peripheral that stops transmission of reception of a busy signal.

When the inserting interface releases its interface, data is output from the first register in its buffer followed by the data in the second register as normal and retaining the order of the data. On the output of the second register data it signals to the previous interface that it is no longer busy and accepts data from the interface.

The previous interface responds to its previous interface in the same fashion, and this control signal ripples back to the source which restarts sending data.

In normal operation only one of the two registers in the buffer are used, the other is dormant.

The header of the data packet includes information relating to a host identifier to indicate the module, an object identifier to indicate the type and name of the data packet and possibly a further object identifier indicating to which of a number of similar objects the data packet belongs.

The bus architecture embodying the present invention uses such identifiers to allow a data packet to be treated as a distinct object that may be applicable to one or more host peripherals. This feature allows a system using these identifiers to be scaled up and down without affecting the bus design. Software using this method can be re-used without knowing the topology of the hardware on which it is working.

Preferably, a flag is associated with the header to determine which of two types it is, a destination or source identifier header. The header type controls how the data packet is treated by the link controller 7.

If the header is a destination type, then the link identifier in the header is used to select the destination host peripheral for the data packet via its associated bus module. When the data packet reaches

its destination, the data is read and is not transferred to the next link.

If the identifier is a source type the link identifier indicates from which host peripheral the payload is being sent or broadcast. If a host peripheral accepts the source and object identifiers in the header as relevant to it, the data packet will be copied and sent onto the next module in the series.

This also provides a mechanism which allows the host peripheral generating the data packet (the source) to confirm transmission of the data packet around the bus. This is accomplished when the source receives a data packet with its own source identifier in the header.

This method also allows a data packet to be sent to multiple host peripherals without having to use multiple data packets as is common in the known bus architectures.

Providing both link and object identifiers in the data packet header allows selectivity based on the type of data and not just on an explicit address. It allows data to be grouped for selective processing. An example of its use in a telecommunications application would be the processing of audio data.

In such a scheme, data from an input source can be routed to multiple host peripherals: a DSP for coding and transmission, a voice recognition module for a command interface, and a memory store if a memo function is required, for example.

In addition such a scheme supports sequential processing of data through a chain of operations contained in the host peripherals.

For example:

Consider an input data packet given a source identifier that relates to a frame in a sequence of

frames and an object identifier that indicates the stage of processing.

Such a data packet would be accepted by a host based on its object identifier, the data would be processed and a new data packet generated with the same source identifier but a new object identifier indicating the processing stage to which the data packet has reached.

This is repeated through the series of host peripherals until it reaches the end of the processing sequence. The last stage uses the source identifier is used to ensure the processed frames remain in order. At any point in the sequence if the process cannot be completed then an error data packet can be generated for transmission to the last processing stage to allow it to resolve the error.

Such a bus structure must ideally be able to be used with legacy peripherals which are based on a standard processor bus architecture. Such a scheme is illustrated in Figure 8. Legacy peripherals can make use of a bus structure embodying the present invention by being placed between two dual port modules. The bus inputs to the peripheral are mounted on one bus and the outputs on the other. In operation they are treated as above using the 'By-Pass' link of the module. Data from one module is input to all standard peripherals. If the identifier is recognised, then it is treated as a simple address, and the interface acknowledges it and blocks the data available signal to the following module thereby preventing acceptance of the data.

If a peripheral addressed in this manner needs to output data then it also forces the busy signal active to the same module to prevent it outputting data on the other bus. It then outputs its data to interface A and indicates the presence of data. The data is maintained

until interface A accepts it, at which point the controlled signals are released.

Some advantages of embodiments of the present invention are presented below:

The standard bus architecture data rate varies between a maximum of the highest speed peripheral and a minimum determined by the slowest peripheral, with a maximum determined by the physical technology of the implementation. In systems with lots of peripherals this results in multiple buses which increases complexity and hence risk and cost.

In contrast, a bus architecture embodying the present invention provides a maximum data rate as a function of the sum total of all data rates of the peripherals. Each link between adjacent modules is limited to a maximum data rate determined by the physical technology but the overall data rate for the bus is not, which leads to a data rate greater than that of a known bus. Also, the physical placement and architecture of embodiments of the present invention should lead to lower complexity and easier testing which reduces risk. Additionally, in embodiments of the present invention, all interfaces to the host peripherals are the same further reducing complexity and testing overheads.

In standard bus architectures the sum loading of all peripherals on the bus limits the maximum speed achievable for a given technology. The higher the load, the higher the drive current required and therefore the higher the power consumed.

In embodiments of the present invention, each module in the bus represents a single load to the previous module. This can be much less than the loading experienced in a standard bus architecture. The number of host peripherals on the bus does not therefore limit the speed of operation due to loading.

Adding more modules to the bus will affect the latency but not the speed of operation.

In standard bus architectures only one peripheral can access another at any given time. As a result the data rate required of a standard bus for a given system is the sum of the total data rates required by each peripheral.

Embodiments of the present invention support multiple parallel communication between host peripherals on different parts of the bus. By placing the peripherals to make use of this the maximum data rate of the system is effectively increased and hence the lifetime of the technology is extended which itself can reduce design costs.

In standard bus architectures when a master unit drives the bus all peripherals are driven; only one address responds but all interfaces have to determine if they are the destination. This results in wasted power consumption for those interfaces which are not addressed.

In embodiments of the present invention, only the host peripherals involved in the communication and the intervening host peripherals are active during data packet transfer, all others are idle and take no part in the transfer. If the bypass link in a module is used in intervening interfaces the activity is further reduced. This can reduce bus activity and therefore power consumption, especially if a self-timed technique is used for implementation.

Software written for standard bus architectures uses absolute addressing and explicit use of the memory locations to identify parameters/objects in the host peripheral. In a new system employing the same peripherals the address may be altered required changes to the software. This requires time for both changes and verification that is costly in most new projects.

Software written for a system using a bus architecture embodying the present invention can be re-used in other systems containing the same host peripherals without modification. This is because the use of object based identifiers in the data packet headers allow parameters and data objects in peripherals to be used without knowing their physical location. This reduces software modifications, and hence cost, and increases re-usability.

In order to reduce latency of a bus architecture embodying the present invention, the modules interfaces between the two modules that require fast access are by-passed the maximum delay being the sum of these small delays. The placement of peripherals can minimise the number of intervening modules such that in such a high-speed mode the latency is equivalent to the standard bus architecture.

Systems using standard buses with two or more processors generally do not allow the driving of peripherals by more than one processor. If a second processor needs to access a peripheral this is achieved by request to the first processor. This reduces the available processing capacity of the first processor for its normal operation, requires specific code on the first processor, and can lead to access conflicts. Such an arrangement can also have a high non-deterministic latency. All of these factors increase the risk, development time and cost of a system.

In contrast, multi-processor systems using embodiments of the present invention allow local use of peripherals associated with a processor but also allow direct access to those peripherals by other processors. No special code is required, and no specific communication between processors is required, since it can be co-ordinated by the shared peripheral. In

addition, latency is more deterministic (and in fact shorter). These factors can reduce risk and cost.

Known system architectures using known buses implement complex data transfer mechanisms when applied to modern telecommunications and related applications. The data flow at the abstract level does not always fit the available data flow at the physical level. This results in wasted resources that limit the performance of the system and hence its lifetime in future applications, thereby increasing cost and development time.

An architecture embodying the present invention supports object and data block based processing which allows easier sequential processing while allowing maximum access to the sequence. This allows more efficient data transfer and flexibility in re-sequencing processes. This then reduces risk in the development while allowing easier addition of future applications while extending the lifetime and scope of the physical technology. The result if lower cost and shorter time to market.

The embodiments also allow broadcast, ie. one-to-many communication.

CLAIMS

1. A bus structure for an integrated circuit, the structure comprising a plurality of interface modules which are connected in series to form the bus structure, each module being operable to transfer data to at least one adjacent module in the series, and including an interface unit for connection with a host peripheral, and a storage element for storing data.

2. A structure as clamed in claim 1, wherein the storage element is operable to store data received from an adjacent module in the series.

3. A structure as claimed in claim 1 or 2, wherein the storage element is operable to store data received from the interface unit.

4. A structure as claimed in claim 1, 2 or 3, wherein the storage element is operable to output data to an adjacent module in the series.

5. A structure as claimed in claim 1, 2 or 3, wherein each module includes a multiplexer which is operable to output data received from one of the interface controller and the storage element.

6. A bus structure as claimed in any one of the preceding claims wherein data transfer between one pair of adjacent modules is independent of data transfer between another pair of adjacent modules.

7. A bus structure as claimed in any one of the preceding claims, wherein each module has an identification value.

8. A bus structure as claimed in any one of the preceding claims, wherein data is transferred along the bus structure in discrete data packets.

9. A bus structure as claimed in any one of the preceding claims, wherein the plurality of interface modules are substantially identical.

10. A bus structure comprising a series of modules, each module comprising:

an input storage element operable to receive data from an adjacent module in the series;

a module controller operable to control data transfer through the module and to exchange control information with an adjacent module in the series; and

an output multiplexer operable to output data to an adjacent module in the series, under the control of the module controller.

The Patent Office
PATENTS • DESIGNS • COPYRIGHT • TRADE MARKS
INVESTOR IN PEOPLE

Application No: GB 0115835.1 **Examiner:** Owen Wheeler
Claims searched: 1-10 **Date of search:** 30 January 2002

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.T): G4A (AFGDC) H4P (PPBC)

Int Cl (Ed.7): G06F: 13/37, 13/40; H04L: 12/42, 12/427, 12/43, 12/433.

Other: Online: EPODOC, JAPIO, WPI

Documents considered to be relevant:

Category	Identity of document and relevant passage		Relevant to claims
X	GB 2188216 A	[PLESSEY] See abstract.	1,10 at least
X	EP 1069509 A2	[TEXAS INSTRUMENTS] See abstract and Figs. 3 and 4.	1,10 at least
X	JP 10-228445 A	[MITSUBISHI ELECTRIC] See abstract and figure.	1,10 at least
X	US 4982400 A	[EBERSOLE] See Figs. 1 and 2.	1,10 at least
X	US 4884192 A	[TERADA] See Figs. 1-3.	1,10 at least
X	US 4641276 A	[DUNKI-JACOBS] See abstract and figure.	1,10 at least

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.